Vers un protocole de routage de surcharge minimale dans les réseaux de capteurs sans-fils.

Henry-Joseph Audéoud, Martin Heusse

Vers un protocole de routage de surcharge minimale dans les réseaux de capteurs sans fils.

Henry-Joseph Audéoud[1] et Martin Heusse[1,2]

[1]*Laboratoire d'Informatique de Grenoble*
[2]*Grenoble INP*

LRP construit un arbre de collecte et des routes descendantes depuis le puits vers les capteurs. Son objectif est de réduire autant que possible le nombre de messages générés pour le routage, parfois au prix de la garantie d'utiliser les plus courts chemins disponibles. De plus, LRP permet la réparation locale de l'arbre et assure à tout moment l'absence de toute boucle de routage.

Cet article détaille et démontre sur la plateforme d'expérimentation FIT IOT-lab deux mécanismes pour réduire encore la surcharge de routage : d'abord un mécanisme d'*expending ring search* lors de la réparation locale, puis l'utilisation opportuniste des messages de construction de l'arbre pour se rapprocher des plus courts chemins.

Mots-clefs : Internet des Objets, routage, réseau de capteurs

1 Introduction

LRP (*Lightweight Routing Protocol*), initialement décrit par [AKHD15], est un protocole de routage à vecteur de distances dédié aux réseaux de capteurs. Il construit proactivement un arbre de collecte pour extraire le trafic hors du réseau. Afin de pouvoir accéder aux capteurs depuis l'extérieur (pour des questions de configuration, de récupération active de données ou simplement pour le renvoi d'acquittements), LRP propose également de construire des routes d'hôtes pour distribuer le trafic à l'intérieur du réseau. Le protocole garantit que le routage se fait sans boucles. Enfin, pour maintenir la communication même en cas de disparition de capteurs ou lorsque les liens radio fluctuent, LRP est pourvu d'un algorithme de réparation locale, qui permet la maintenance de l'arbre sans qu'il soit nécessaire d'en réparer globalement la structure.

L'approche choisie pour LRP est d'aller vers un protocole aussi silencieux que possible, au prix éventuellement de l'utilisation garantie des routes les plus courtes, sachant qu'il est toujours possible ensuite de dédier plus de ressources pour contrecarrer les pertes de paquets de routage, les collisions *etc.* Cet article présente deux améliorations à LRP allant dans le sens d'une diminution de la surcharge de routage pour des routes les plus courtes possible. La première fonctionnalité permet de réduire drastiquement le nombre de messages utilisés lors de la réparation locale de la structure de l'arbre. La seconde présente une manière d'optimiser la construction de l'arbre, tout en limitant le nombre de messages envoyés. Ces deux propositions sont accompagnées d'expériences sur IOT-lab qui illustrent leur efficacité.

2 LRP — Présentation du protocole

2.1 Arbre de collecte

L'arbre de collecte (ou plus exactement le graphe orienté sans cycle dirigé vers le routeur de bordure du réseau, appelé le *puits*) définit un ensemble de routes par défaut, permettant de collecter efficacement le trafic. Sur ce point, LRP est plus proche de RPL [WTB+12] que de LOADng-CTP [YCdV12] qui suppose que les préfixes distribués au sein du réseau de capteurs sont disjoints.

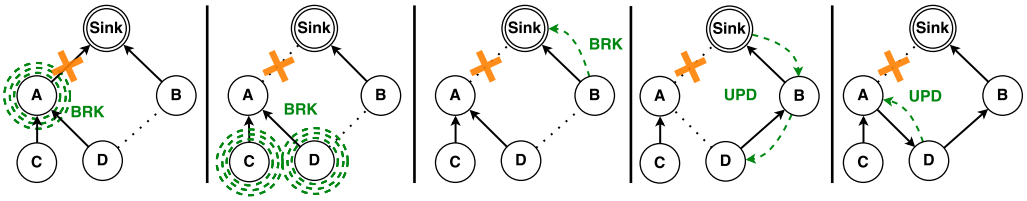

FIGURE 1: Mécanisme de réparation locale

L'arbre est construit grâce à l'algorithme de Bellman-Ford distribué : chaque nœud associé au réseau diffuse un message appelé *DODAG Information Object* (DIO) autour de lui ; les nœuds du voisinage peuvent alors le sélectionner comme *successeur*, c'est-à-dire leur *next-hop* sur la route par défaut.

En plus de la métrique nécessaire au fonctionnement de l'algorithme, les DIO possèdent un numéro de séquence. Celui-ci permet de dater chronologiquement l'arbre, et d'effectuer une réparation globale (incrément du numéro de séquence, ce qui entraîne la reconstruction de l'arbre à partir de zéro).

RPL utilise l'algorithme Trickle pour diffuser des messages à l'intérieur du réseau. LRP laisse ce choix à l'implémentation : le coût en énergie de transmission des messages, surtout des *broadcasts*, n'est pas le même suivant la plateforme et les radio utilisées.

2.2 Routes d'hôtes

LRP propose de construire des routes d'hôtes afin de distribuer le trafic IP dans le réseau. La construction d'une route peut se faire de manière proactive, à la façon de RPL, à l'initiative de chaque nœud ; toutefois, contrairement à RPL, les routes peuvent également être établies à l'initiative du puits, de manière réactive, comme le fait LOADng. L'acheminement du trafic est garanti sans boucles de routage : celles-ci sont détectées et supprimées dès qu'elles apparaissent [AKHD15].

2.3 Réparation locale

L'algorithme de Bellman-Ford nous permet d'avoir un arbre de plus courts chemins, mais peut engendrer des phases de comptage à l'infini quand le coût d'un lien augmente ou qu'un relais disparaît. Comme dans tous les protocoles de routage à vecteur de distances augmentés d'un numéro de séquence, LRP impose aux nœuds de ne jamais † augmenter leur distance au puits pour un numéro de séquence donné. Par contre, LRP inclut un mécanisme pour réparer l'arbre avec peu de trafic de routage lorsqu'un capteur se trouve déconnecté.

Ce mécanisme de réparation locale, ou LR (*Link Reversal*), est déclenché lorsqu'aucun voisin ne peut être utilisé comme successeur sans éloigner le nœud déconnecté du *puits*. Ayant épuisé les possibilités de réparation « vers le haut », le nœud cherche alors une solution *via* ses prédécesseurs.

Dans la figure 1, le lien entre A et le puits qui avait servi à construire l'arbre n'est plus utilisable. A diffuse dans l'ensemble de son sous-arbre un message BRK (BReaK). Lorsqu'un relais potentiel reçoit ce message d'un nœud qui n'est pas son successeur (il se trouve alors à l'extérieur de l'arbre, et n'est donc pas directement influencé par la perte du lien), il le remonte en *unicast* au puits. Celui-ci répond par un message de type UPD (UPDate) qui parcourt en sens inverse — et en unicast — le chemin suivi par le BRK. À l'instar des DIO, un UPD permet aux nœuds de choisir leur successeur. D choisi ainsi B comme nouveau successeur, puis A choisi D. Le lien entre A et D est renversé par rapport à la situation initiale, et la route par défaut entre A et le puits est rétablie.

3 Réduction de la surcharge du routage

3.1 Utilisation d'expending ring search lors de la réparation locale

Lors d'une réparation locale, le message BRK est diffusé dans tout le sous-arbre du nœud ayant initié la réparation locale. Or, parmi tous les chemins de réparation possible, les plus courts d'entre eux pour

†. RPL permet d'augmenter la distance au puits mais en bornant l'éloignement pour un numéro de séquence donné.

rejoindre le puits sont ceux passant par les prédécesseurs les plus proches. Afin de diminuer le nombre de messages émis il est donc intéressant de limiter la diffusion du BRK à ces nœuds-là. L'algorithme classique d'*expending ring search* nous permet d'obtenir un tel résultat.

Lorsqu'un nœud initie une réparation locale, il spécifie dans le BRK un champ *ring_size*, décrivant la distance maximale que le BRK est autorisé à parcourir dans le sous-arbre. Initialement, ce champ vaut 0 ; il est incrémenté à chaque re-génération d'un BRK, en l'absence de réponse. Les relais du sous-arbre décrémentent ce champ avant chaque retransmission. Lorsqu'il devient nul, le message n'est pas rediffusé en *broadcast* plus loin. Toutefois, il peut encore être retransmis en *unicast* en direction du puits : il est alors en dehors du sous-arbre, le long de l'arbre de collecte encore fonctionnel, sa retransmission ne dépend plus de la taille du *ring*.

3.2 Réponse spontanée à un DIO

Afin de réduire la surcharge du routage, il est important de tirer de chaque message toute l'information disponible. Ainsi, lorsque le voisin d'un nœud A à une distance n du puits diffuse un DIO décrivant une distance moins bonne que $n+1$, A répond spontanément en renvoyant un message DIO *unicast* décrivant sa propre distance, incitant de cette façon le voisin à changer de successeur.

Grâce à ce mécanisme, un DIO contenant une distance infinie remplace un DIS (*DODAG Information Sollicitation*), car il a le même rôle : solliciter l'envoi en retour d'un DIO.

Cette amélioration permet d'avoir tout d'abord, à faible coût, un arbre plus proche de l'arbre minimal, réduisant par là la surcharge de routage et d'acheminement du trafic. Ensuite, elle permet à un nœud de sonder son environnement pour détecter des successeurs potentiels, sans déclencher une avalanche de réponses, comme cela aurait été le cas s'il avait utilisé un DIS : seuls les DIO pouvant l'intéresser lui seront transmis.

Il est à noter qu'un DIO n'est transmis en *broadcast* que lorsque les nœuds modifient leur position dans l'arbre (association au réseau, rapprochement du puits ou changement de numéro de séquence) ou lors de la réparation de l'arbre.

4 Résultats pratiques

Afin de vérifier nos résultats par la pratique, nous avons réalisé ces deux propositions, pour les tester sur la plateforme FIT IoT-lab ‡. L'arbre de collecte obtenu apparaît figure 2. Le nœud 34 a une sensibilité augmentée, ce qui lui permet de se connecter directement au puits, et servir ainsi de relai aux autres nœuds (38 et 39). Après 4 minutes et demie, le nœud 34 est éteint, ce qui force les nœuds 38 et 39 à trouver une route alternative pour atteindre le puits.

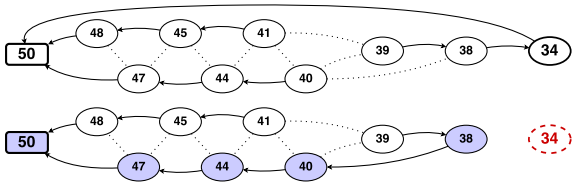

FIGURE 2: Arbre de collecte du réseau LRP lors de l'expérience. Le nœud 50 est le puits. Les flèches représentent l'arbre de collecte ; les traits en pointillés représentent des liens inutilisés. Au-dessus : situation après convergence de l'algorithme de Bellman-Ford ; en-dessous, après réparation locale, suite à l'extinction du nœud 34. Les nœuds sur fond sombre sont ceux ayant vu passer le message UPD de réparation.

Dans la figure 3, à 6 minutes, les DIO sont diffusés par le nœud 38 après détection de la perte de son successeur. Il n'y a pas de réponse : aucun nœud dans son voisinage ne peut lui fournir une route aussi courte que celle que lui fournissait 34. 38 initie alors une réparation locale, en diffusant un BRK. Comme le champ *ring_size* est à 0, celui-ci n'est pas retransmit par 39, mais seulement en *unicast* par 40 et 41. Le puits répond par un UPD, et l'arbre de collecte est réparé.

‡. `www.iot-lab.info`

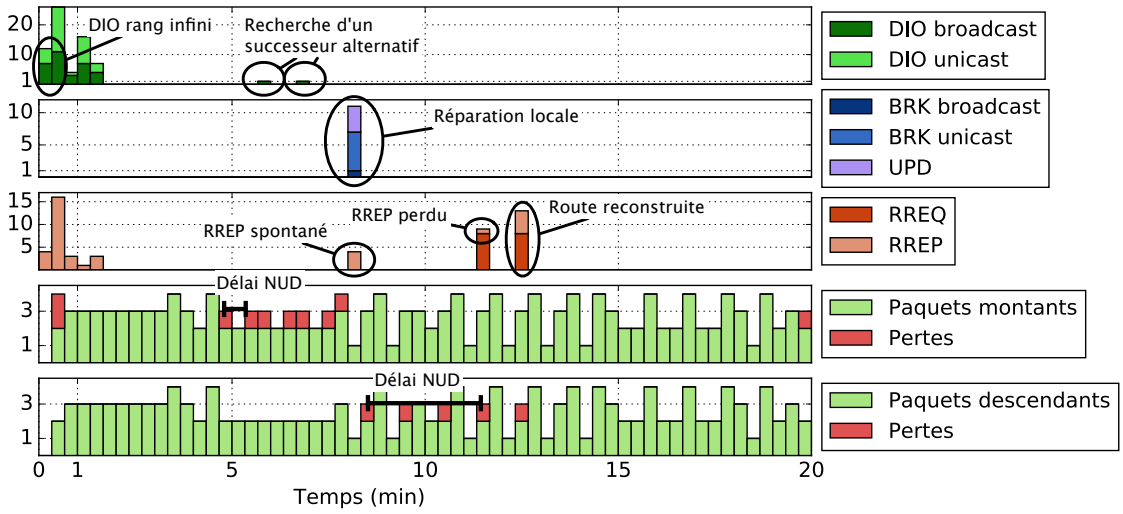

FIGURE 3: Messages échangés lors de l'expérience. Les nœuds sont initialisés entre 0 et 30". À 4'30", le nœud 34 servant de relai à d'autres nœuds est éteint. La connexion est complètement rétablie à 12'30".

Toutefois, toutes les routes ne sont pas encore rétablies. En effet, 38 a pu envoyer un RREP spontané pour mettre à jour sa route d'hôte, car il est au courant de la réparation ; 39 n'a pas vu passer l'UPD, et n'a pas agi de même, la route obsolète passant par 34 n'a donc pas été mise à jour. Ceci entraine des pertes de paquets dans le sens descendant, jusqu'à ce que la route d'hôte soit rétablie de façon réactive, grâce à des RREQ.

5 Conclusion

Les mécanismes présentés ci-dessus montrent que la recherche d'un protocole de routage aussi silencieux que possible est encore un problème ouvert. Nous pensons que LRP doit permettre de concevoir des protocoles adaptés à des contextes divers, en partant d'un protocole parcimonieux, validé sur plateforme réelle en parallèle de la simulation.

Beaucoup d'autres optimisations sont encore possible. Par exemple, une fois la réparation locale terminée, les routes d'hôtes déservant le sous-arbre réparé ne sont pas forcément mis à jour. Un mécanisme permettant de rafraîchir proactivement ces routes d'hôtes permettrait d'éviter leur reconstruction réactive, et par là une inondation du réseau.

LRP est disponible à `https://github.com/drakkar-lig/contiki.git`, branche « lrp ».

Références

[AKHD15] Henry-Joseph Audéoud, Michal Krol, Martin Heusse, and Andrzej Duda. Low overhead loop-free routing in wireless sensor networks. In *WiMob 2015*, pages 443–451. IEEE, Oct 2015.

[PNP+10] Duy Ngoc Pham, Van Duc Nguyen, Van Tien Pham, Ngoc Tuan Nguyen, Xuan Bac Do, Trung Dung Nguyen, C. Kuperschmidt, and T. Kaiser. *An expending ring search algorithm for Mobile Adhoc networks*, page 39–44. IEEE, Oct 2010.

[WTB+12] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J-P. Vasseur, and R. Alexander. RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), March 2012.

[YCdV12] Jiazi Yi, T. Clausen, and A.C. de Verdiere. Efficient data acquisition in sensor networks : Introducing (the) loadng collection tree protocol. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pages 1–4, Sept 2012.